

חוברת עזר

אלקטרוניקה ותכנות

תחרות לאומית רחפנים-תשפ"ג



כתב: אבי חיון

תוכן עניינים

3.....	מבוא	1
3.....	1.1 תיאור סכמתי של הכרטיס המורכב על הרחפן	
5.....	מבוא ל- Arduino	2
7.....	הגדרת כניסות ויציאות דיגיטליות	3
7.....	3.1 כתיבת להדק דיגיטלי	
8.....	3.2 קריאת ערך הכניסה מהדק דיגיטלי	
9.....	4 מבוא לפרוטוקולי תקשורת בין רכיבים	
10.....	5 תקשורת טורית UART (Universal Asynchronous Receiver Transmitter)	
10.....	5.1 פרוטוקול תקשורת	
11.....	5.2 תקשורת UART ב- Arduino	
13.....	6 תקשורת טורית אלחוטית באמצעות רכיבי Bluetooth	
14.....	6.1 תכנית דוגמה	
15.....	7 חיישן Ultrasonic	
16.....	7.1 מאפייני החיישן	
16.....	7.2 עיקרון פעולת החיישן	
18.....	7.3 חיישן מרחק Ultrasonic Sensor 6 מטר	
19.....	7.4 פונקציה pulseIn	
19.....	7.5 תוכנית דוגמה	
20.....	8 NeoLed נורת LED מסוג WS2812	
20.....	8.1 פרוטוקול העברת המידע	
21.....	8.2 ספרייה Adafruit_NeoPixel	
22.....	8.3 תכנית דוגמה	
24.....	9 מנוע RC-Servo	
25.....	9.1 מבנה הפנימי של מנוע SERVO	
26.....	9.2 פונקציות ה- SERVO	
26.....	9.3 תכנית דוגמה	
27.....	9.4 חיבור המנוע	
28.....	10 נגד רגיש לאור LDR - Light Dependent Resistors	
30.....	11 תקשורת טורית בפרוטוקול I2C (Inter-Integrated Circuit)	
31.....	11.1 מבנה פרוטוקול התקשורת	
32.....	11.2 ספרייה Wire	
34.....	12 רכיב הפועל בפרוטוקול I2C	
34.....	12.1 חיישן צבע TCS34725	
34.....	12.1 מבוא- צבע	

35.....	תיאור הרכיב TCS34725	12.2
36.....	ספריית Adafruit_ TCS34725	12.3
36.....	תכנית דוגמה לזיהוי הצבע	12.4
39.....	כללים לבניית פרויקט	12.5
39.....	שלבנים ל- debug בפרויקט	12.6
40.....	מקורות	13
41.....	נספח	14
41.....	סביבת הפיתוח ב-Arduino	14.1

1 מבוא

חוברת זו, מטרתה לספק רקע עיוני ובסיס למידה במערכות האלקטרוניות השימושיות בתחרות הרחפנים. כחלק מהכרת המערכות תיחשפו הן לתחום החומרה והן לתחום התוכנה, לרבות, תכניות דוגמה להפעלת הרכיבים, עקרונות פיסיקליים והסבר עיקרון הפעולה של הרכיבים השונים.

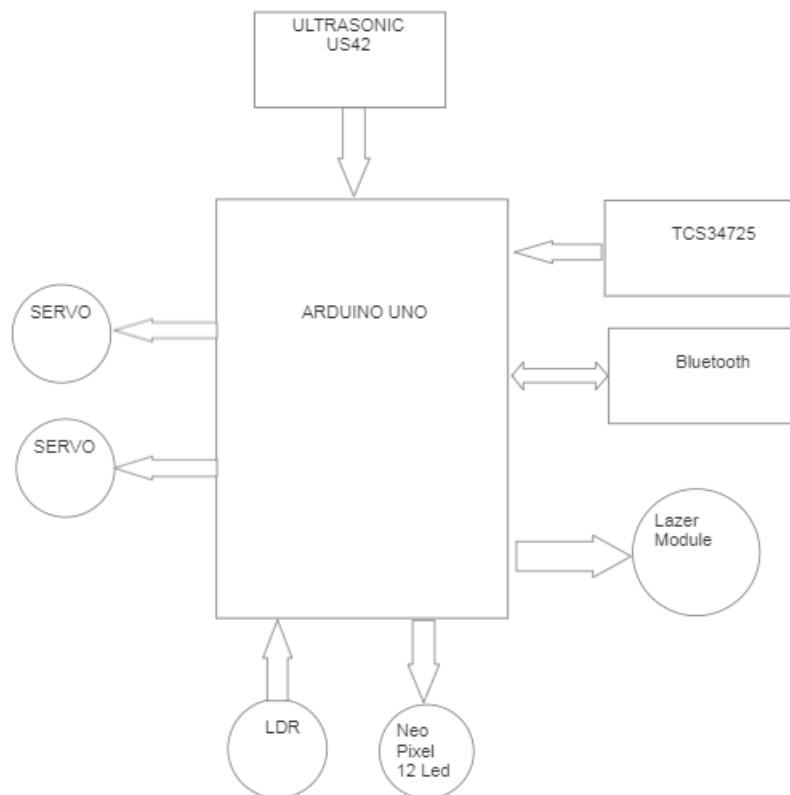
המערכות מתחלקות לשני חלקים עיקריים:

1. **כרטיס הרחפן**, חיישנים שונים ורכיבי הפעלה שונים המורכבים סביב Arduino uno.

2. **אפליקציה** המותקנת על טאבלט/פלאפון.

התקשורת בין הכרטיס הנ"ל לטאבלט/פלאפון, תהיה אלחוטית באמצעות רכיב Bluetooth.

1.1 תיאור סכמתי של הכרטיס המורכב על הרחפן



הכרטיס מורכב מהיחידות הבאות:

1. רכיב Arduino uno – משמש כבקר המערכת על פי תוכנה בשפת C++
2. חיישן צבע – משמש לקביעת ערכי צבע בסביבת החיישן .
3. חיישן מרחק Us42 – משמש למדידת מרחק עד 6 מטר.
4. חיישן LDR המשמש לקליטת עוצמת האור בסביבתו.
5. שני מנועי סרבו – מיועד לפתיחת וו הגורם להפלת כדורים מהרחפן.
6. מערכי LED צבעונים WS2812 מעגל של 12 לדים – יורכבו בתחתית הרחפן מתחת לכל מנוע.
7. משדר/ מקלט Bluetooth HC06 – משמש לתקשורת עם יחידת המשתמש, אפליקציה.

2 מבוא ל- Arduino

Arduino היא חברה איטלקית שהוקמה ב-2005 ומפתחת סביבת פיתוח ולוחות, ביניהם מיקרו-בקרים מסדרת AVR של חברת ATMEGA המיועדים לשימוש בתוכניות חינוכיות שונות בעולם כולו ובישראל.

סביבת הפיתוח

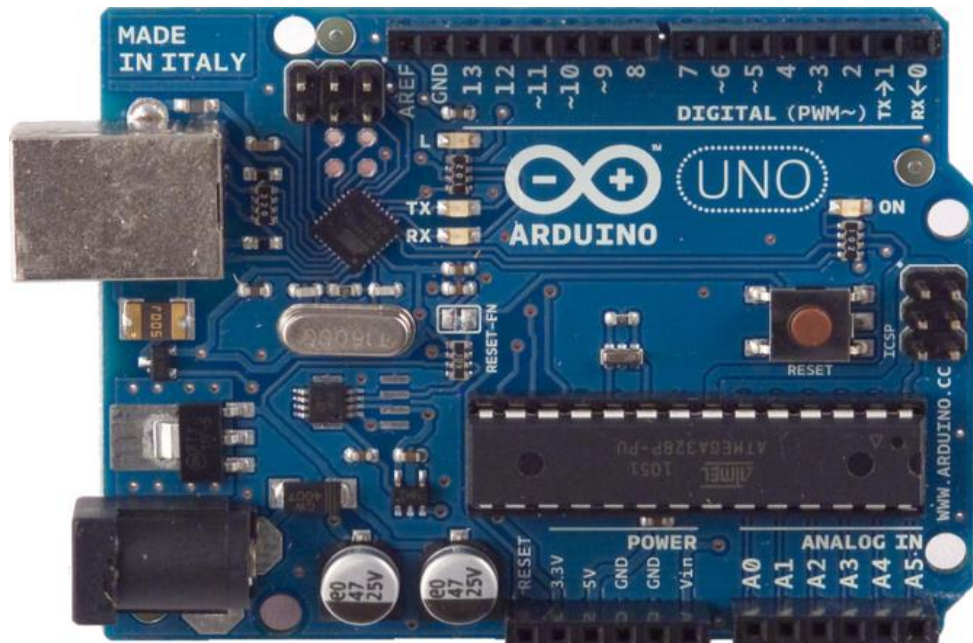
סביבת הפיתוח של Arduino קלה וידידותית לשימוש, הכוללת שפע דוגמאות וספריות ובהן פרויקטים עם קוד פתוח שהשימוש בהם מקצר את זמן הפיתוח. הספריות והפונקציות מאפשרות לבנות קודים המסוגלים להגיב או לשלוט על חיישנים של חום, קול, מגע, אור ותנועה. בעזרת הקודים והחיישנים ניתן ליצור מגוון של פרויקטים דוגמת רובוטים, משחקים, כלי נגינה.

חומרה

לוחות הפיתוח והחיישנים

חומרת Arduino כוללת מגוון רחב של לוחות ובהתאם ללוח ניתן למצוא בקרים שונים החומרה מתחברת בצורה נוחה וניתן להוסיף Arduino Shields (כרטיסים נלווים לכרטיס הראשי) לוחות שתוכננו לביצוע משימה ייחודית, וניתן לחברם ישירות ללוח ה-Arduino באמצעות פינים מוארכים בחלקם העליון.

לוח Arduino Uno



זהו המפרט הטכני של הלוח:

- מיקרו-בקר Atmega328
- מתח עבודה 5v למעבד (מתח אספקה לכרטיס נעים בתחום (7 ~ 12).
- מחבר USB לצריבת קוד לבקר וחיבור מתח לכרטיס.
- עשרים כניסות ויציאות דיגיטליות - מתוכן שש יציאות PWM ושש כניסות אנלוגיות
- זרם בהדקי I/O עד 40mA
- תדר שעון 16MHZ
- לחצן Reset
- ארבע נורות LED המשמשות לחיוויים אלה:
 - Led On - משמשת לחיווי מתח הכניסה.
 - Led Tx - משמשת לחיווי שליחת נתון בתקשורת טורית.
 - Led Rx - משמשת לחיווי קבלת נתון בתקשורת טורית.
 - Led L - מחוברת להדק 13 לתרגול.

3 הגדרת כניסות ויציאות דיגיטליות

ההדקים של המיקרו-בקר יכולים לשמש ככניסות או יציאות דיגיטליות (יציאה המוציאה מתח של '1' לוגי (5V) או '0' לוגי (0V)). לכן, לפני השימוש בהם עלינו להגדירם למצב כניסה או יציאה. פעולה זו תתבצע באמצעות פונקציה pinMode, אשר מקבלת שני פרמטרים:

- פרמטר ראשון, מציין את מספר ההדק של לוח ה- Arduino.
- פרמטר שני, קובע את מצב ההדק לאחד משלושת המצבים הבאים:
 - INPUT - כניסה.
 - INPUT_PULLUP - כניסה עם נגד המחובר למתח 5V.
 - OUTPUT - יציאה.

להלן פורמט הפונקציה:

pinMode (מצב ההדק, מספר ההדק);

3.1 כתיבת להדק דיגיטלי

פעולה זו מתבצעת באמצעות פונקציה digitalWrite, אשר מקבלת שני פרמטרים:

פרמטר ראשון קובע את מספר ההדק בלוח ה- Arduino.

פרמטר שני קובע את מצב הלוגי של ההדק, בשני אופנים:

- רמת לוגית של '0' לוגי = LOW = 0.
- רמת לוגית של '1' לוגי = HIGH = 1

להלן דוגמה לתוכנית הגורמת ללד המחובר להדק 13 להבהב כל שנייה

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off
  delay(1000);                      // wait for a second
}
```


3.2 קריאת ערך הכניסה מהדק דיגיטלי

פעולה זו מתבצעת באמצעות פונקציה `digitalRead` המקבלת פרמטר המציין את מספר ההדק שאנו מעוניינים לקרוא. הפונקציה מחזירה ערך 0 או 1, ולכן כדאי להגדיר משתנה מטיפוס הכי קטן `char/byte` על מנת לחסוך בזיכרון. משתנה זה ישמור את הערך המוחזר. להלן דוגמה לתוכנית הבודקת את מצב הלחצן, במידה והוא לחוץ, ה-LED המחובר להדק 13 דולק.

```
#define buttonPin 2 //the number of the pushbutton pin
#define ledPin 13 // the number of the LED pin

// variable for reading the pushbutton status
int buttonState = 0;

void setup() {
    // initialize the LED pin as an output:
    pinMode(ledPin, OUTPUT);
    // initialize the pushbutton pin as an input:
    pinMode(buttonPin, INPUT);
}

void loop() {
    // read the state of the pushbutton value:
    buttonState = digitalRead(buttonPin);
    /* check if the pushbutton is pressed. If it is, the
    buttonState is HIGH:*/
    if (buttonState == HIGH)
        digitalWrite(ledPin, HIGH); // turn LED on:
    else
        digitalWrite(ledPin, LOW); // turn LED off:
}
```

4 מבוא לפרוטוקולי תקשורת בין רכיבים

פרוטוקול תקשורת בין רכיבים הוא אוסף של כללים המגדירים את התקשורת בין רכיבים טוריים. כאשר הרכיב שיוזם את התקשורת ומספק את אות השעון נקרא Master והרכיב שאליו פונים נקראה Slave.

לרוב ה-Master הוא מיקרו-בקר שלנו הוא Atmega328 (המיקרו-בקר שעל Arduino Uno) ורכיב ה-Slaves הם רכיבים פריפריאליים כמו חיישנים. רוב הפרוטוקולים כוללים את הכללים הבאים:

- תחילת תקשורת
- העברת מידע
- אימות / תיקון שגיאות (לא חובה)
- סיום תקשורת.

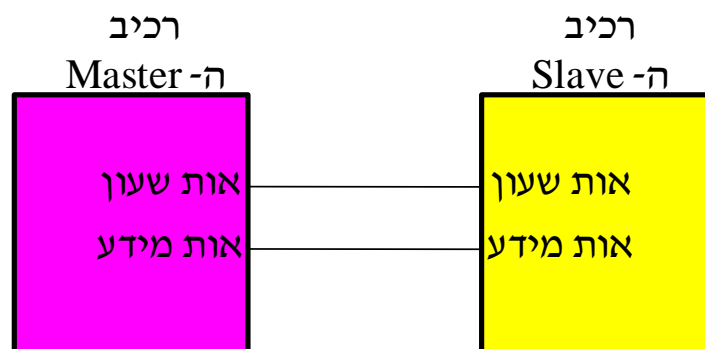
בחוברת זו נתמקד בשני פרוטוקולים טורים UART ו-I2C שיוסברו בהמשך.

פרוטוקול תקשורת טורית

בפרוטוקול זה המידע נשלח סיבית אחר סיבית בין רכיב ה-Master ורכיב ה-Slave כמות הסיביות שמעוברות והסדר השליחה שלהם יכול להיות שונה. ניתן לחלק את הפרוטוקול הזה גם לתקשורת טורית סינכרונית או אסינכרונית.

פרוטוקול תקשורת טורית סינכרונית

בפרוטוקול זה אות השעון משותף בין רכיב ה-Master ורכיב ה-Slave והמידע מעוברת סיבית אחר סיבית בהתאם לאות השעון.



פרוטוקול תקשורת טורית אסינכרונית

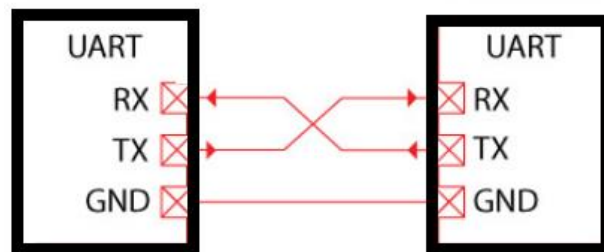
בפרוטוקול זה אות השעון אינו משותף בין רכיב ה-Master ורכיב ה-Slave והמידע המועבר מחולק למסגרת מידע (frames), זמן השידור מוגדר מראש. כאשר לפני שידור שולחים סיבית תחילת שידור Start bit ובסיום שולחים סיבית סיום Stop Bit.

5 תקשורת טורית UART (Universal Asynchronous Receiver Transmitter)

זהו פרוטוקול תקשורת טורית בין שני התקנים בעל המאפיינים הבאים:

- תקשורת אסינכרונית (ללא הדק שעון CLK).
- המידע יכול לעבור בו זמנית בין ה- MASTER ל-SLAVE - Full Duplex.
- פועל עם שני קווים בלבד – TXD, RXD.
- קצב אופייני עד 460K ביטים בשנייה.
- תקשורת בין Master אחד ל- Slave אחד.

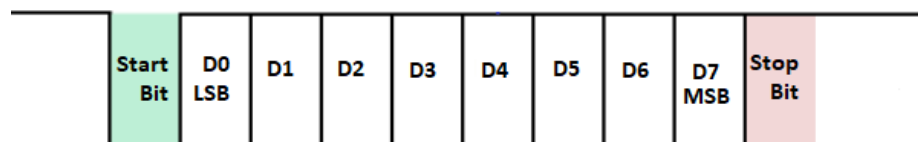
בדרך כלל מועבר מידע של 8 סיביות החל מ-D0 (LSB) עד D7 (MSB), למידע זה מתווספת סיבית התחלה לתזמון בין המקלט למשדר, וסיבית סיום המודיעה על סיום שידור של המידע.



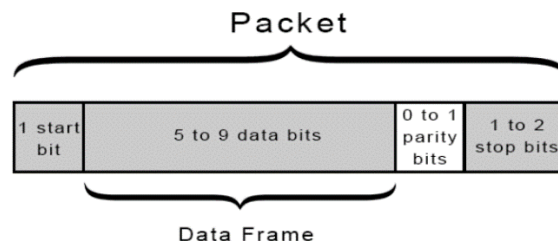
5.1 פרוטוקול תקשורת

המידע משודר בקצב של מספר הביטים בשנייה, המוגדר כ- Baud Rate

משך הזמן של כל ביט מחושב כ- $1/\text{Baud Rate}$



חבילת המידע יכולה להשתנות לפי האיור הבא:



המידע יכול להיות בין 5 ל-9 סיביות בתוספת של סיבית זוגיות לבדיקת שגיאות ועד שני סיביות סיום. שימוש נפוץ שידור של 8 סיביות וסיבית סיום אחת.

5.2 תקשורת UART ב-Arduino

פונקציות פלט בתקשורת הטורית

פונקציה print

באמצעות פונקציה זו ניתן לשלוח מחרוזות (תווים בקוד ASCII) מהמיקרו-בקר של לוח Arduino אל מחשב ה-PC או התקן אחר המכיל UART.

פונקציה println

פונקציה זו מבצעת את פעולת הפונקציה print אך בנוסף מורידה את הסמן לתחילת השורה הבאה (n).

להלן דוגמה לשליחת ערך משתנה בתקשורת טורית והצגתו על המסך:

```
void setup() {
  //initialize serial communication at 9600 bits per second
  Serial.begin(9600);
}

byte count = 0 ;
void loop() {
  // print out the value count:
  Serial.println(count++);
  delay(1000);    //delay 1sec
}
```

פונקציות קלט

פונקציות אלו משמשות לקליטת נתונים בתקשורת טורית ממחשב ה-PC או מכל התקן העובד בתקשורת טורית UART. הנתונים שנקלטים בתקשורת הטורית מאוחסנים ב-buffer העובד בשיטת FIFO (First In First Out) וגודלו 64Bytes. יש מספר פונקציות הקשורות לקבלת נתונים.

פונקציה available (זמין)

פונקציה זו מחזירה את מספר התווים שהתקבלו בתקשורת טורית. התווים שהתקבלו מאוחסנים ב-Buffer העובד עם יחידת FIFO וגודלו 64Bytes. במידה שב-buffer לא מאוחסן נתון, הפונקציה תחזיר את הערך 0. דוגמה לקריאת נתון בתקשורת הטורית ושליחה בחזרה:

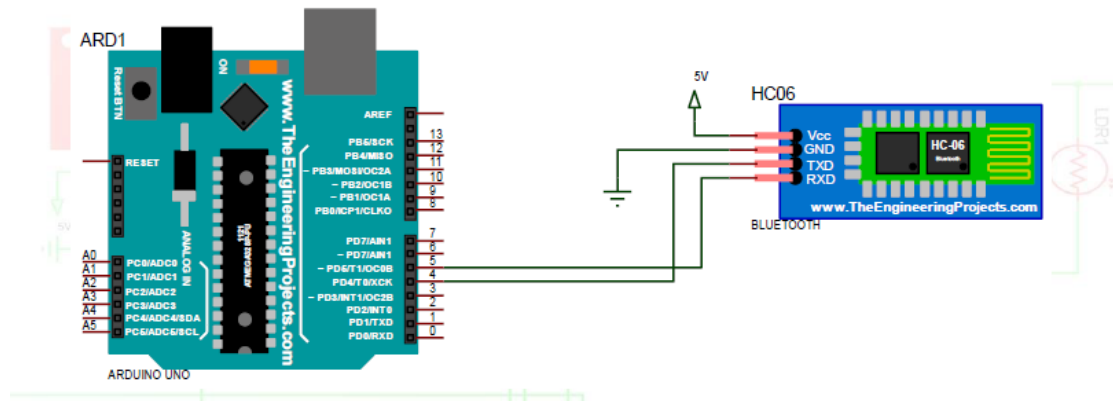
```
void setup() {
  //initialize serial communication at 9600 bits per second
  Serial.begin(9600);
}

void loop() {
  byte dataSerial;
  // if there's any serial available, read it:
  if (Serial.available() > 0) {
    dataSerial = Serial.read(); //read data from UART
    Serial.print(dataSerial);   // send data to UART
  }
}
```

6 תקשורת טורית אלחוטית באמצעות רכיבי Bluetooth

תקשורת טורית בפרוטוקול UART אפשרית הן בצורה חוטית (כפי שהוסבר בפרק הקודם) והן בצורה אלחוטית על ידי שתי יחידות מקמ"ש (משדר/ מקלט) או שני רכיבי Bluetooth.

האיור הבא מתאר תקשורת אלחוטית באמצעות רכיב Bluetooth המחובר לארדואינו לבין יחידת ה- Bluetooth הקיים בטאבלט.



התקשורת האלחוטית היא דו כיוונית, רכיב Bluetooth HC05 הממוקם על הארדואינו משמש כ-MASTER והרכיב השני הממוקם בטאבלט משמש כ-SLAVE. ה-MASTER יוזם את חיבור התקשורת האלחוטית באמצעות שם וקוד המשותף לשני רכיבי Bluetooth. מרגע ההתחברות קיימת תקשורת דו כיוונית ללא חשיבות מי ה-MASTER. התקשורת פעילה כל עוד המרחק סביר בין היחידות.

הסבר מפורט על רכיבי Bluetooth ותכונותם מתואר באתר הבא:

<http://www.arikporat.com/arduino1/hc-05%20bluetooth.pdf>

בדוגמה הנ"ל חוברו רכיבי הבלוטוס להדקי D4, D5 של הארדואינו שאינם הדקי ה-UART של הבקר, לכן יש צורך להגדירם לצורת תקשורת הנקראת SoftwareSerial - תקשורת בתוכנה.

לצורך שימוש בתקשורת זו, יש להגדיר ספרייה

```
#include <SoftwareSerial.h>
```

להגדיר אובייקט ולקבוע את הדקי התקשורת

```
SoftwareSerial mySerial(4, 5); // D4 => RX , D5 => TX
```

כמו כן, קביעת קצב התקשורת, פונקציות שידור וקליטה דומים לתקשורת Serial המובנת של הארדואינו.

6.1 תכנית דוגמה

להלן תכנית לשידור וקליטת byte אחד בתקשורת טורית בין שני רכיבי ארדואינו,

באמצעות SoftwareSerial

תוכנית המשדרת ערך מספרי בגודל byte מ-0 עד 255 בצורה מחזורית

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(4, 5); // RX, TX
void setup() {
  mySerial.begin(9600);
}
byte cnt=0;
void loop() {
  delay(500);
  mySerial.write(cnt++);
}
```

תוכנית הקולטת נתון אחד ומציגה אותו על המסך הסריאלי

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(4, 5); // RX, TX
void setup() {
  mySerial.begin(9600);
  Serial.begin(9600);
  Serial.println("Serial Test");
}
void loop() {
  if (mySerial.available()) {
    byte num = mySerial.read();
    Serial.println(num);
  }
}
```

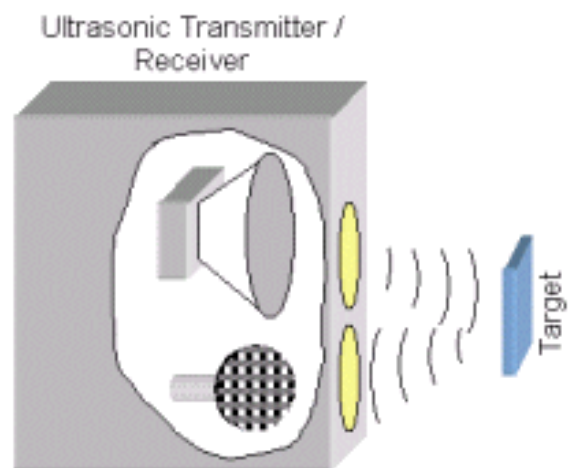
7 חיישן Ultrasonic

חיישן Ultrasonic מיועד למדוד את המרחק של גוף הנמצא מולו.

החיישן בנוי ממשדר ומקלט, הוא משדר גלי קול בתדר 40KHz הנעים במהירות של 343 מטר לשנייה, גלי קול אלה פוגעים בגוף וחוזרים למקלט.

הזמן העובר בין תחילת השידור לקליטת הגל החוזר, הוא למעשה הזמן שלוקח לאות לעבור פעמים את המרחק לעצם (הלוך וחזור).

על פי זמן זה ומהירות הקול, ניתן לחשב את המרחק.



מבנה וחיבור החיישן



לחיישן 4 הדקים בלבד, דבר המפשט את חיבורו למערכת.

VCC - מתח הפעלה.

GND - אדמה.

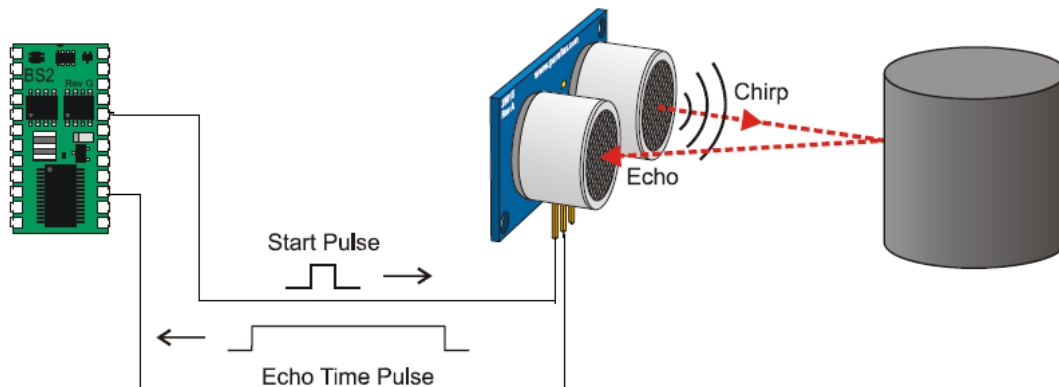
Trigger - הדק כניסה להפעלת המשדר.

Echo - הדק יציאה המספק אות היחסי למרחק.

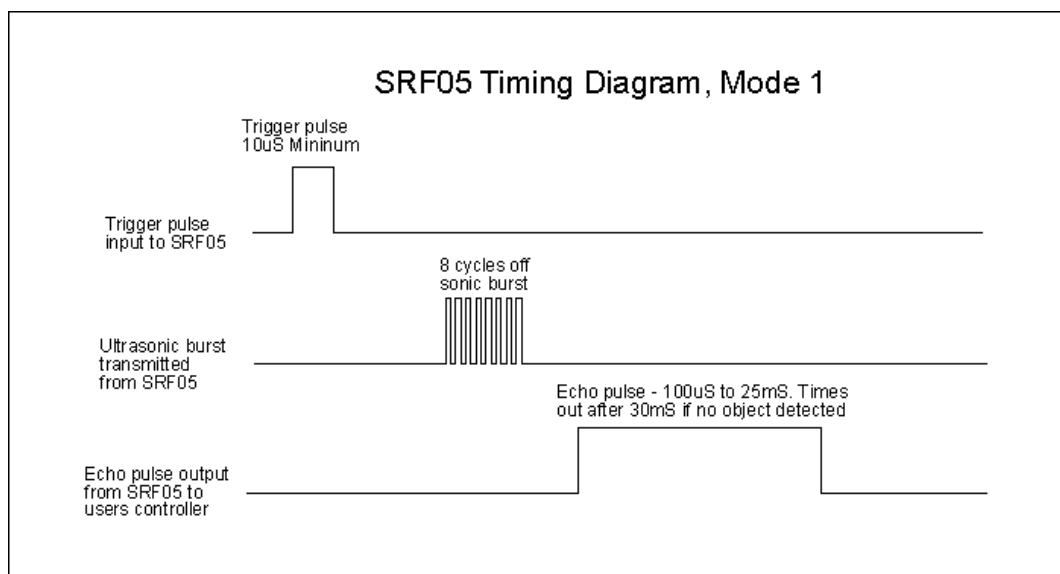
7.1 מאפייני החיישן

- מתח הפעלה - 5 וולט
- זרם במצב מנוחה קטן מ- 2 mA.
- זרם עבודה 50mA .
- תדירות גל קול – 40KHz .
- טווח פעולה – 2 ס"מ עד 4 מטר.
- זווית פעולה – 30 מעלות, טווח יעיל עד 15 מעלות.
- רזולוציה - 0.3 ס"מ.

7.2 עיקרון פעולת החיישן



תרשים גלים של החיישן



בתחילת הפעולה, מספקים להדק ה- Trigger של החיישן פולס חיובי ברוחב מינימלי של 10 מיקרו שנייה, דבר הגורם למשדר של החיישן לחולל 8 מחזורים של גלי קול בתדירות 40kHz.

מקלט החיישן ימתין לגל החוזר מהעצם ויספק בהדק היציאה Echo, פולס חיובי ברוחב היחסי למרחק כפי שמתואר בתרשים הנ"ל.

במידה ולא קיים עצם מול החיישן, המתח בהדק הסה Echo ירד כעבור 25msec.

המדידה

גלי הקול הנשלחים על ידי החיישן נעים במהירות הקול, השווה בערך ל-343 מטר לשנייה.

אם לדוגמא ישנו עצם במרחק של 1 מטר מהחיישן, גלי הקול הנשלחים יפגעו בו ויחזרו אל החיישן, כך שהמרחק הכולל הוא 2 מטר.

רוחב הפולס המתקבל בהדק הסה Echo מייצג את הזמן שעבר מיציאת גל הקול ממשדר החיישן ועד לקליטתו ע"י המקלט מחדש.

ערכו שווה לדרך שעבר גל הקול חלקי מהירות הקול לפי הנוסחה הבאה:

$$t = \frac{x}{v} = \frac{2m}{343m/s} \cong 5.8msec$$

ניתן לראות שאם עצם נמצא במרחק של 1 מטר מהחיישן, רוחב הפולס בהדק הסה Echo יהיה שווה ל- 5.8msec.

אם כך, עבור כל 1 ס"מ נקבל זמן של 58μs.

באמצעות התוכנה נמדוד את רוחב הפולס ברגל Echo ביחידות μs ונמיר אותה לתוצאה בס"מ.

7.3 חיישן מרחק Ultrasonic Sensor 6 מטר

חיישן Ultrasonic מיועד למדוד את המרחק של האובייקט הנמצא מולו והוא בנוי ממשדר ומקלט ומשדר גלי קול בתדר 40KHz. ניתן לקבוע את mode העבודה של החיישן לאחת משלוש האפשרויות הבאות:

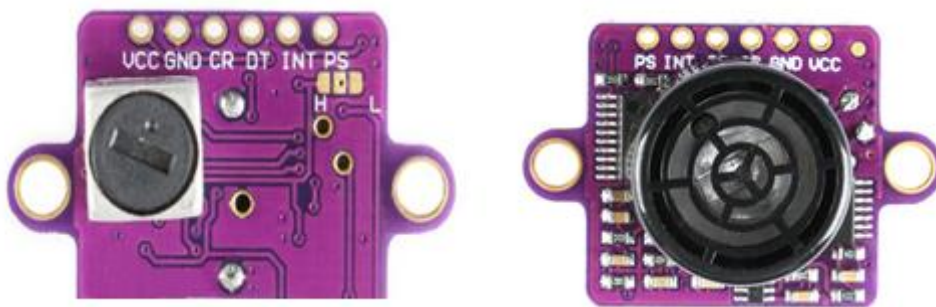
1. Mode עבודה של מדידת רוחב פולס.

2. Mode עבודה לפרוטוקול טורי UART.

3. Mode עבודה לפרוטוקול טורי I2C.

במידה ועובדים ב-mode עבודה של רוחב פולס, פעולת Trigger היא פנימית ומתקבלת מהמיקרו-בקר של החיישן.

להלן תמונת חיישן וחיבור הדקים



פירוט הדקים של החיישן ב-mode עבודה של רוחב פולס:

הדק VCC - הדק לחיבור מתח של 5V.

הדק GND - הדק לחיבור להדק GND.

הדק CR - הדק לא בשימוש.

הדק DT - הדק המשמש למוצא מהחיישן למדידת רוחב הפולס.

הדק INT - הדק לא בשימוש.

הדק PS - הדק לקביעת mode העבודה של החיישן למדידת רוחב פולס.

שימו לב! ב-mode עבודה הנוספים ההדקים יהיו בשימוש.

קביעת mode עבודה של החיישן למדידת רוחב פולס מתבצעת חיבור הדק PS ל-GND

('0' לוגי). ניתן לבצע פעולה זו ע"י הלחמה ההדק PS להדק L.

למדידת רוחב הפולס המתקבל מהחיישן נשתמש בפונקציה pulseIn

7.4 פונקציה `pulseIn`

באמצעות פונקציה זו ניתן למדוד אורך פולס ברמה של '1' לוגי או ברמה של '0' לוגי. הפונקציה מחזירה את הערך הנמדד (ביחידות מיקרו) במידה שיש שינוי בפולס, ובמידה שאין שינוי בפולס היא מחזירה את הערך 0.

7.5 תוכנית דוגמה

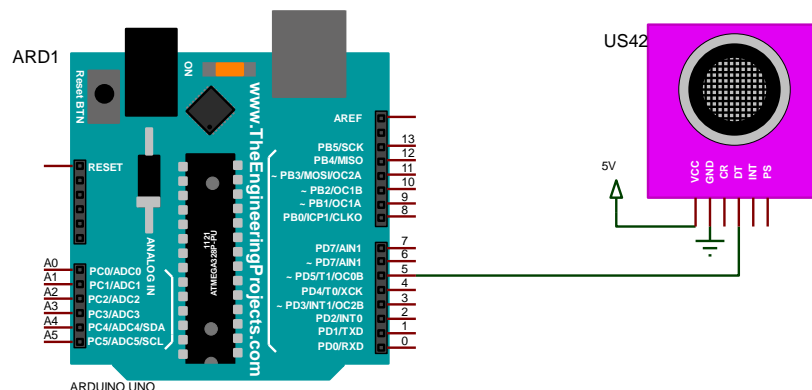
להלן דוגמה לקריאה ערך מהחיישן והצגתו על גבי המסך:

```
#define echoPin 3

void setup() {
  Serial.begin(9600);
  pinMode(echoPin, INPUT);
}

void loop() {
  word duration, distance;
  /* pulseIn() waits for the pin to go from LOW to HIGH,
   starts timing, then waits for the pin to go LOW and
   stops timing.
   Returns the length of the pulse in microseconds*/
  duration = pulseIn(echoPin, HIGH);
  distance = duration / 58; // 1cm give 58 microseconds
  Serial.print("distance = ");
  Serial.println(distance);
  delay(50);
}
```

להלן סכמה החשמלית לחיבור החיישן אל לוח Arduino:



8 NeoLed נורת LED מסוג WS2812

נורה המכילה שלוש נורות LED בשם RGB LED (Red, Green, Blue) בגודל 5x5mm הכוללת מעגל משולב המאפשר לשלוט בעוצמת ההארה של כל נורה באמצעות שיטת אפנון רוחב פולס PWM (Pulse Width Modulation) בגודל שמונה סיביות (256 רמות שונות). התקשורת עם הנורה מתבצע באמצעות פרוטוקול טורי WS2812 הכולל שליחת 24 סיביות מידע המכילות את ערך הצבע שיש לכתוב לכל נורה. בסיום שליחת המידע מבצעים פעולת start code הטוענת את המידע לנורות. מתח העבודה של הנורה הוא 5V וניתן לשרש מספר רב של נורות.

זהו צילום מסך של נורת ה-LED



פירוט הדקי הנורה:

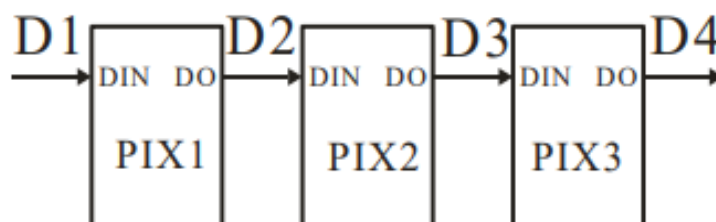
NO.	Symbol	Function description
1	VDD	Power supply LED
2	DOUT	Control data signal output
3	VSS	Ground
4	DIN	Control data signal input

8.1 פרוטוקול העברת המידע

כל מעגל LED RGB משורשר אחד לשני, מוצא DO של הראשון מחובר לכניסה DIN של השני וכך הלאה.

המידע עובר בצורה טורית לפי האיור הבא:

Cascade method:



24 סיביות הראשונות המרכיבות את הצבע של ה-LED מועברות למעגל ה-LED הראשון,
24 סיביות הבאות מועברות ל-LED2 וכך לפי סדר מעגלי ה-LED.

המידע הטורי לכל נורית LED מכיל 24 סיביות (8 עבור LED ירוק, 8 עבור האדום ו-8 עבור הכחול).

Composition of 24bit data:

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

מידע זה מומר לאפנון PWM לכל LED, כך שנקבל 256 מצבי עוצמה שונים לכל LED.
אפנון PWM מאפשר לווסת את עוצמת ההארה של ה-LED על ידי שינוי ה- Duty Cycle של הגל הריבועי וכך לשינוי הערך הממוצע של המתח המסופק ל-LED.
וויסות עוצמת ההארה של שלושת ה-LED מאפשר קביעת צבע ההארה של RGB LED בגווי הצבע השונים.

8.2 ספרייה Adafruit_NeoPixel

ספרייה זו מאפשרת להפעיל נורות RGB LED WS2812B. כדי לעבוד עם ספרייה זו יש להצהיר על שימוש בספרייה Adafruit_NeoPixel על ידי שורת הקוד הבאה:

פונקציה begin

באמצעות פונקציה זו מאתחלים את ספריית neoPixel.

פונקציה setPixelColor

באמצעות פונקציה זו ניתן לכתוב לזיכרון של הנורה. הפונקציה מקבלת ארבעה פרמטרים:

פרמטר ראשון מציין את מספר הנורה שרוצים לפנות.

פרמטר שני מציין את ערך הצבע של הנורה שצבעה אדום.

פרמטר שלישי מציין את ערך הצבע של הנורה שצבעה ירוק.

פרמטר רביעי מציין את ערך הצבע של הנורה שצבעה כחול.

פונקציה show

באמצעות פונקציה זו ניתן להפעיל נורות RGB LED WS2812B . הפונקציה כותבת את ערך השמור בזיכרון אל הנורות.

פונקציה clear

באמצעות פונקציה זו מאפסים את ערכים של הנורות בזיכרון גודל הזיכרון שמאפסים נקבע בהתאם לערך מספר הנורות.

8.3 תכנית דוגמה

להלן תכנית המדליקה כל חצי שניה LED אדום רץ ממערך של 16 LEDs, לאחר מכן בצבע ירוק וכחול.

```
#include <Adafruit_NeoPixel.h>
//Which pin on the Arduino is connected to the NeoPixels
#define PIN          A0
// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS    16

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS,
PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  pixels.begin();
  pixels.clear();
}

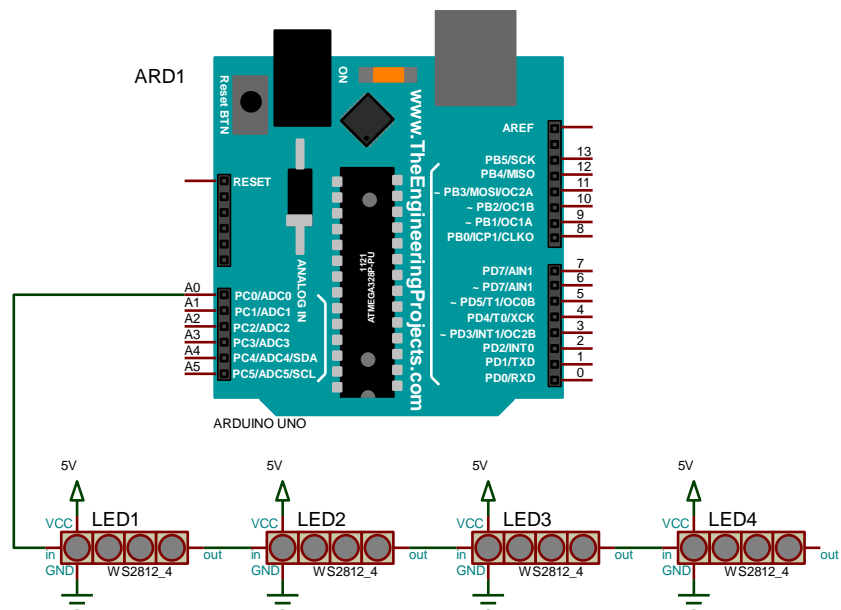
void loop() {
  for (int i = 0; i < NUMPIXELS; i++) {
    pixels.setPixelColor(i, 100, 0, 0);
    pixels.show();
    delay(500);
  }
  pixels.clear();  pixels.show();
}
```

```

delay(500);
for (int i = 0; i < NUMPIXELS; i++) {
    pixels.setPixelColor(i, 0, 100, 0);
    pixels.show();
    delay(500);
}
pixels.clear();  pixels.show();
delay(500);
for (int i = 0; i < NUMPIXELS; i++) {
    pixels.setPixelColor(i, 0, 0, 100);
    pixels.show();
    delay(500);
}
pixels.clear();  pixels.show();
delay(500);
}

```

להלן סכמה החשמלית לחיבור מערך ה-LEDs אל לוח Arduino:
ניתן לחבר את ארבעת בטבעות בטור או כל טבעת להדק אחר, יש להתייחס לכך בתוכנה.



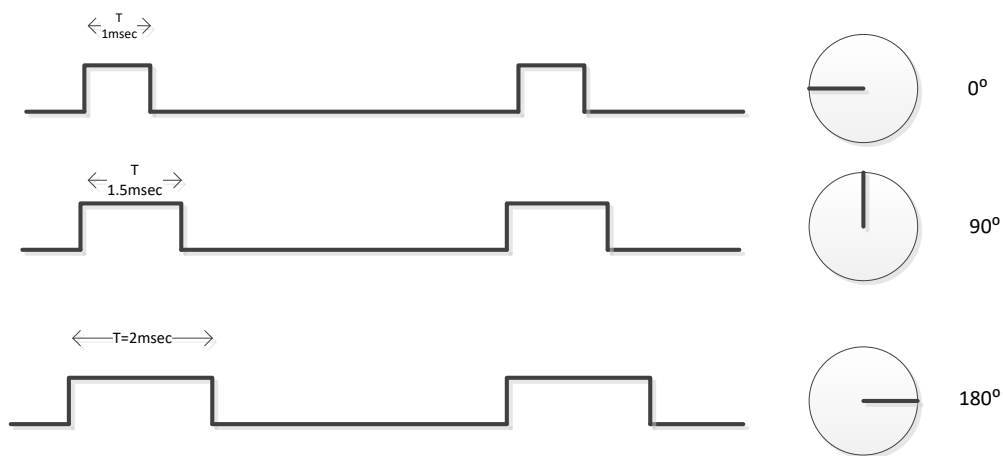
9 מנוע RC-Servo



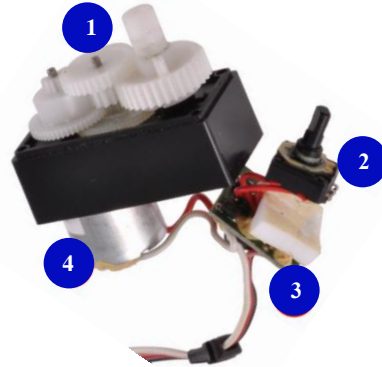
מנוע סרבו כולל מנוע זרם ישר (DC MOTOR) בעל מערכת תמסורת פנימית של גלגלי שיניים ובקרה אלקטרונית על מיקום ציר המנוע.

מנועי סרבו אינם מסתובבים בצורה חופשית אלא נעים לאורך זווית שלרוב בין 0 ל-180 מעלות. בגלל יתרונות רבים שיש למנועי סרבו (קלות השליטה-רמת מתח TTL וממשק נוח עם מיקרו-מעבדים, מומנט גבוה, משקל וגודל נמוכים, יעילות-צריכה נמוכה של אנרגיה). הם גם נפוצים מאוד בישומי רובוטיקה ומשמשים להנעה של רובוטים ניידים קטנים וזרועות רובוטיות.

הבקרה היא ע"י פולס דיגיטלי בכניסה ברוחב T הקובע את זווית המנוע.



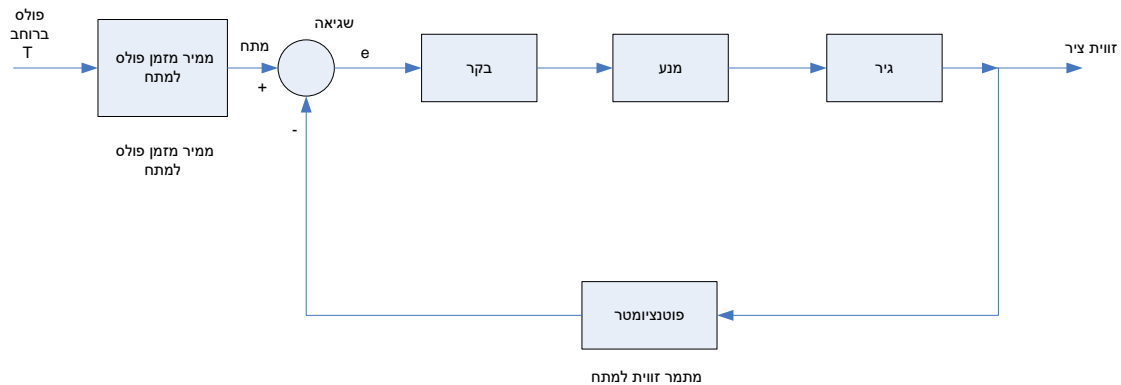
9.1 מבנה הפנימי של מנוע SERVO



מצילום המסך לעיל ניתן לראות את הרכיבים הבאים:

1. תמסורת מכאנית עושיה מפלסטיק או מתכת (יותר מומלץ) עם או בלי מסבים.
2. פוטנציומטר לקביעת מיקום המנוע.
3. מעגל אלקטרוני להפעלת ולבקרה על המנוע.
4. מנוע DC עם מברשת.

סכמת בלוקים המתארת את הבקרה על המנוע



הסבר- זהו מעגל בקרה בחוג סגור. אות הכניסה הוא פולס ברוחב T הקובע את הזווית הרצויה. אות זה מומר למתח אנלוגי היחסי לרוחב T , מתח זה משווה עם מתח הפוטנציומטר היחסי לזווית הציר של מנוע ה-SERVO. במידה וקיימת שגיאה מתבצע תיקון (סבוב המנוע לכוון המתאים) עד למצב שהשגיאה שווה 0 (עצירה).

אות הכניסה ברוחב T בתחומים בין $(700 \sim 2300) \mu\text{sec}$ מסופק על ידי בקר Arduino בצורה מחזורית.

9.2 פונקציות ה-SERVO

ספריית Servo שמוכנת בסביבה העבודה

ספרייה זו משמשת להפעלת מנועי Servo בשפת תוכנת Arduino.

פונקציה `attached`

פונקציה זו בודקת האם הדק היציאה מחובר למנוע Servo, במידה וכן, הפונקציה מחזירה `true` במידה ולא, הפונקציה תחזיר `false`.

פונקציה `attach`

מגדיר את הדק החיבור של מנוע ה-Servo.

פונקציה `write`

באמצעות פונקציה זו ניתן לכתוב ערך למנוע כדי לשנות את זווית צירו. הערך הנשלח יהיה בתחום $0 \sim 180$.

להלן דוגמה להגדרת זווית ציר המנוע שמחובר להדק 4 לזווית של 90°

```
myservo.attach(4);
```

```
myservo.write(90); // set servo 90°
```

פונקציה `writeMicroseconds`

באמצעות פונקציה זו ניתן להגדיר זווית למנוע על ידי שליחת ערך רוחב הפולס ביחידות μsec .
להלן דוגמה להגדרת זווית מנוע Servo שמחובר להדק 4 ל- 90° (רוחב פולס שך $1500\mu\text{sec}$
מביא את המנוע לאמצע - 90°)

```
myservo.attach(4);
```

```
myservo.writeMicroseconds(1500); // set servo 90°
```

9.3 תכנית דוגמה

תכנית הגורמת לסיבוב המנוע מ-0 עד 180 כל 15 מילישניות, זמן הסיבוב 2.7 שניות

($15\text{msec} \times 180$) וחזרה מ-180 ל-0 מעלות .

```
#include <Servo.h>
```

```
Servo myservo; // create servo object to control a servo
```

```
int pos = 0; // variable to store the servo position
```

```

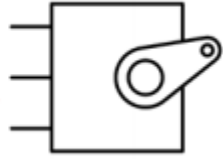
void setup() {
  // attaches the servo on pin 4 to the servo object
  myservo.attach(4);
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    myservo.write(pos); //turn from 0 to 180 degrees
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo.write(pos); // turn from 180 to 0 degrees
    delay(15);
  }
}

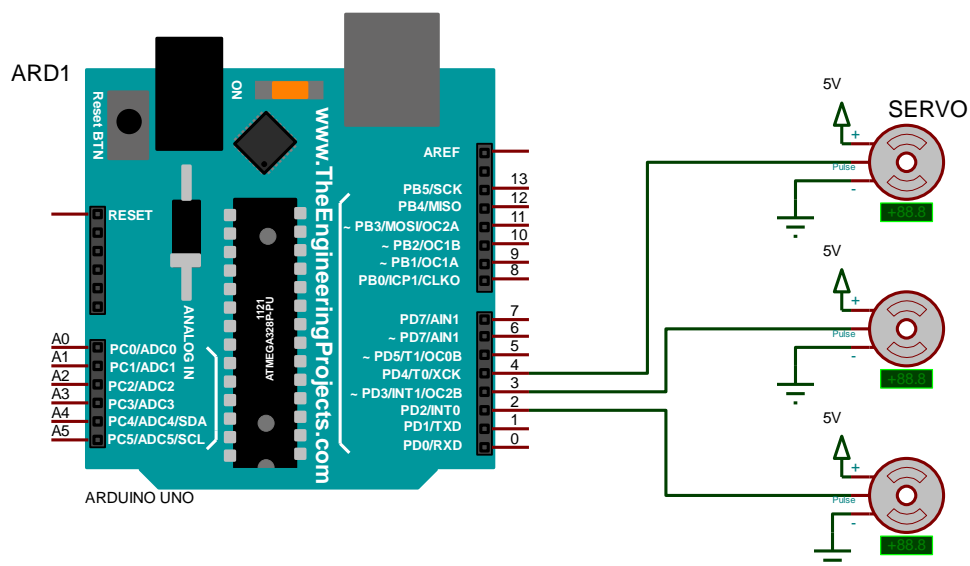
```

9.4 חיבור המנוע

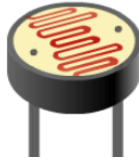
PWM=Orange (⏏)
 Vcc = Red (+)
 Ground=Brown (-)



להלן שרטוט חשמלי המתאר את חיבור שלושת המנועים אל לוח Arduino:



10 נגד רגיש לאור LDR - Light Dependent Resistors

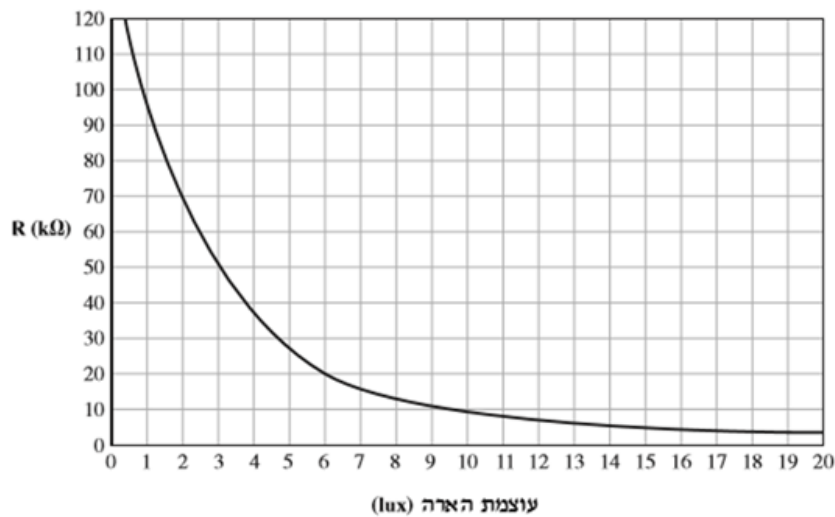


זהו חיישן שהתנגדותו משתנה בהתאם לעוצמת אור הסביבה. ככל שעוצמת האור עולה ההתנגדות יורדת.

התנגדות ה-LDR אינה משתנה באופן ליניארי כפונקציה של השינויים בעוצמת האור.

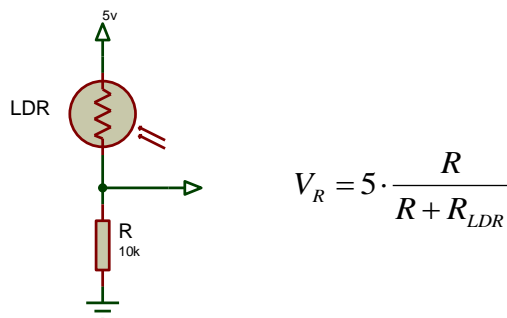
אחד החומרים שבהם משתמשים ליצירת נגד רגיש לאור הוא קדיום סולפיד (Cadium Sulphide).

הגרף הבא מתאר את הקשר בין התנגדות החיישן לעוצמת ההארה:



על מנת שנוכל לקרוא את עוצמת ההארה באמצעות בקר הארדואינו, יש צורך להמיר את שינוי ההתנגדות לשינוי מתח ולאחר מכן להמיר את המתח למידע דיגיטלי באמצעות ממיר (ADC (Analog to Digital Conversion

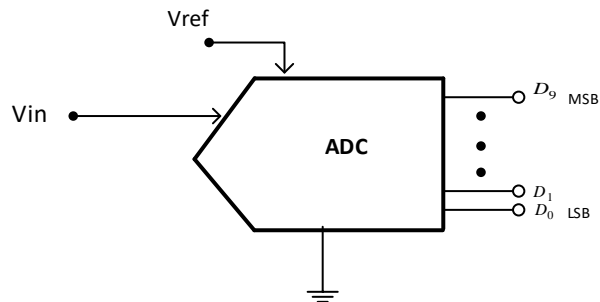
המרת ההתנגדות למתח נעשית באמצעות חיבור נגד בטור לחיישן לפי המעגל הבא:



לפי הנוסחה הנ"ל מתקיים קשר בין המתח להתנגדות החיישן, לכן ככל שעוצמת האור גדלה, התנגדות החיישן קטנה ולכן מתח הנגד גדל.

קריאת המתח האנלוגי באמצעות בקר האדואינו

בתוך בקר האדואינו קיים ממיר אנלוגי לדיגיטלי ADC בגודל 10 סיביות אשר מספק ערך בין 0 ל- $2^n - 1$ 1023



הקשר בין הערך הספרתי במוצא לבין מתח הכניסה הוא:

$$Data = \frac{Vin}{\Delta V} = \frac{Vin}{V_{REF} / 2^n}$$

ΔV - רזולוציית (אבחנה) הממיר בוולטים

$-V_{REF}$ מתח הייחוס של הממיר (ברירת מחדל באדואינו 5V , ניתן לשינוי)

n – מספר סיביות , 10

Vin – מתח הכניסה של הממיר, יכול להיות מכל כניסה אנלוגית של האדואינו A0 עד A5

לאחר הצבה בנוסחה נקבל :

$$Data = \frac{Vin}{\Delta V} = \frac{Vin}{5/2^{10}} \approx \frac{Vin}{4.883mV}$$

תוכנית דוגמה לקריאת ערך המתח האנלוגי מכניסה A0 והצגתו במסך הסריאלי

```
const int ldrPin = A1; // Analog input pin connect to LDR
int ldrValue ;

void setup() {
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  ldrValue = analogRead(ldrPin);

  // print the results to the Serial Monitor:
  Serial.print("LDRsensor = ");
  Serial.println(ldrValue);

  delay(1000);
}
```

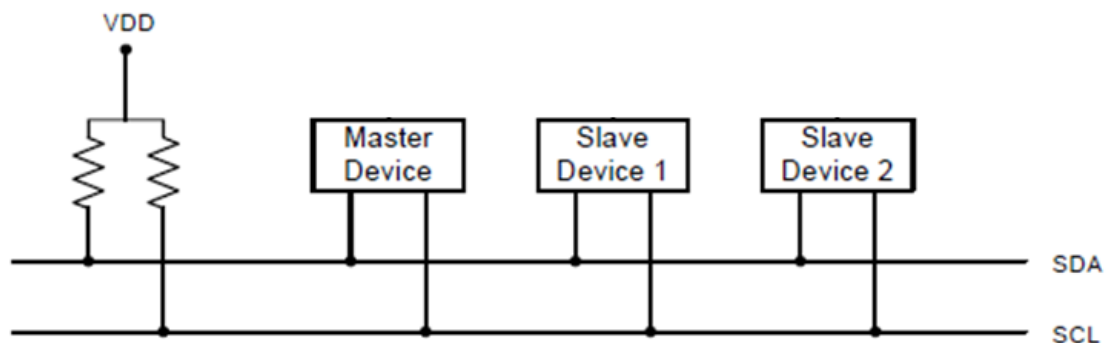
11 תקשורת טורית בפרוטוקול I2C (Inter-Integrated Circuit)

פרוטוקול תקשורת טורית I2C הוא פרוטוקול של BUS (קווי חיבורים בין היחידות לעברת מידע ואותות), בעל שני חוטים בלבד לתקשורת עם נתונים בקצב איטי עד בינוני. הפרוטוקול פותח על ידי חברת Philips בתחילת שנות השמונים והשימושים הראשונים בו היו ברכיבים פנימיים של טלוויזיה ורדיו.

מאפייני התקשורת:

- תקשורת סינכרונית (כולל שעון CLK).
- התקשורת ברגע נתון מתבצעת בין ה-Master לאחד ה-Slave בלבד, בכיוון אחד בלבד - Half Duplex.
- אפשרי לחבר מספר התקני SLAVE בעלי כתובת שונה.
- בעל שני קווי תקשורת (SCL, SDA).
- קצב אופייני עד 400K ביטים בשנייה.

חיבור התקשורת טורית סינכרונית בין התקן Master לבין התקני Slave



פס התקשורת I2C מורכב משני קווים:

SDA - Serial Data, קו מידע דו כיווני.

SCL – Serial Clock, קו שעון מה-Master ל-Slave.

לכל רכיב המחובר ל-BUS ישנה כתובת בגודל שבע סיביות ייחודית, כאשר הוא מקבל אותה בהדקי הכתובות שלו, הרכיב מופעל. כל רכיב המחובר ל-BUS יכול גם לשלוח מידע וגם לקבל (כיוון שה-BUS הוא דו כיווני) אבל לא באותו הזמן.

11.1 מבנה פרוטוקול התקשורת

Start(S)	7 Bits Address	Read/ Write Bit	ACK/ NACK Bit	8 Bits Data	ACK/ NACK Bit	8 Bits Data	ACK/ NACK Bit	Stop(P)
----------	----------------	-----------------------	---------------------	-------------	---------------------	-------------	---------------------	---------

סדר הפעולות בפרוטוקול התקשורת I2C

1. התחלת התקשורת, ה- MASTER שולח START(S).
2. MASTER שולח כתובת הרכיב בגודל 7 סיביות + סיבית R/W.
3. SLAVE בעל הכתובת שולח אישור 'ACK=0' (ACKNOWLEDGE).
4. MASTER קורא או כותב 8 ביטים .
5. סיום התקשורת, ה- MASTER שולח STOP (P) .

ספריית Wire שמובנית בסביבת העבודה של Arduino

ספרייה זו מפעילה את יחידת TWI (Two-Wire Interface) של הבקר ומאפשרת שימוש נוח בפרוטוקול I2C

הספרייה מכילה פונקציות שונות. כדי לעבוד איתן יש להצהיר על שימוש בקובץ כותרת Wire.h. הדקי הבקר של Arduino המשמשים לתקשורת זו הם:

- הדק A5 - הדק שעון SCL .
- הדק A4 - הדק מידע SDA.

פונקציה **Wire.requestFrom**

באמצעות פונקציה זו ניתן לקרוא נתון או נתונים מרכיב ה-Slave והיא מקבלת שני פרמטרים: הפרמטר הראשון מציין את כתובת רכיב ה-Slave שעמו רוצים לתקשר. והפרמטר השני קובע את מספר הנתונים בגודל Byte שרוצים לקרוא.

פונקציה **Wire.available**

מחזירה את מספר הנתונים שנקראו מרכיב ה-Slave.

פונקציה **Wire.read**

באמצעות פונקציה זו ניתן לקרוא נתון בגודל Byte מרכיב ה-Slave.

דוגמה לקריאת נתון

להלן תוכנית לקריאת נתון מתקשורת i2c מרכיב Slave בכתובת 8 והצגתו על גבי מסך התקשורת הטורית.

```
#include <Wire.h>

void setup() {
    Wire.begin(8);           // slave device #8
    Serial.begin(9600);      // start serial for output
}

void loop() {
    while (Wire.available()) {
        char c = Wire.read(); // receive a byte as character
        Serial.print(c);      // print the character
    }
    delay(500);
}
```

12 רכיב הפועל בפרוטוקול I2C

12.1 חיישן צבע TCS34725

12.1 מבוא- צבע

האור הוא גל אלקטרומגנטי המורכב מספקטרום רחב, חלק מהספקטרום האלקטרומגנטי נראה לעין בצורת צבעים שונים לפי אורך הגל שאורכו בין 350 ל-750 ננומטר. אורך הגל הקצר ביותר שעינינו מבחינות בו נקרא "סגול", והארוך ביותר נקרא "אדום". האור הפוגע בעין, מורכב מגלים רבים שאורכי הגל שלהם שונים. העין מאבחנת את התערובת בתור צבע מסוים, הנקרא גוון. אפשר ליצור צבעים אחרים ע"י חיבור של צבעים או חיסור.

תכונות הצבע

כדי לאפיין צבע מסוים משתמשים בשלוש תכונות:

גוון - מייצג את אורך הגל של הצבע.

כרומטיות - מייצג את חוזק החומר ותלוי בחלק היחסי של הצבע הטהור.

בהירות – מייצג את עוצמת האור המוחזרת מהגוף. ההבדל בין הצבע הלבן לבין דרגות שונות של אפור הוא בבהירות.

חיישן צבע – Color Sensor

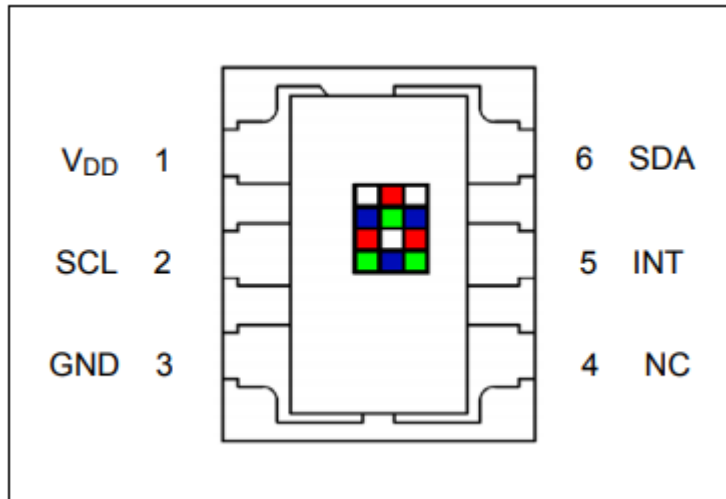
כוללים מסננים לחסימת אור IR לא רצוי בטווח ספקטרלי גלוי, המאפשר מדידת צבע מדויקת מאוד. לאור הצבע יש אורכי גל שונים, אלה מזוהים להמרה לרמות הנוכחי על ידי גלאי צילום ואז האות מעובד. לבסוף האות ברמה הנוכחית מומר לערכים דיגיטליים על ידי ADC (אנלוגי ממיר דיגיטלי). נתוני התפוקה יכולים להיות מטופלים על ידי מעבד או מיקרו בקר כדי לענות על היישומים שלנו.

12.2 תיאור הרכיב TCS34725

הינו חיישן צבע בעל רגישות גבוהה, טווח דינמי רחב וכולל מסנן חסימת IR הנותן פתרון לשימוש בתנאי תאורה שונים.

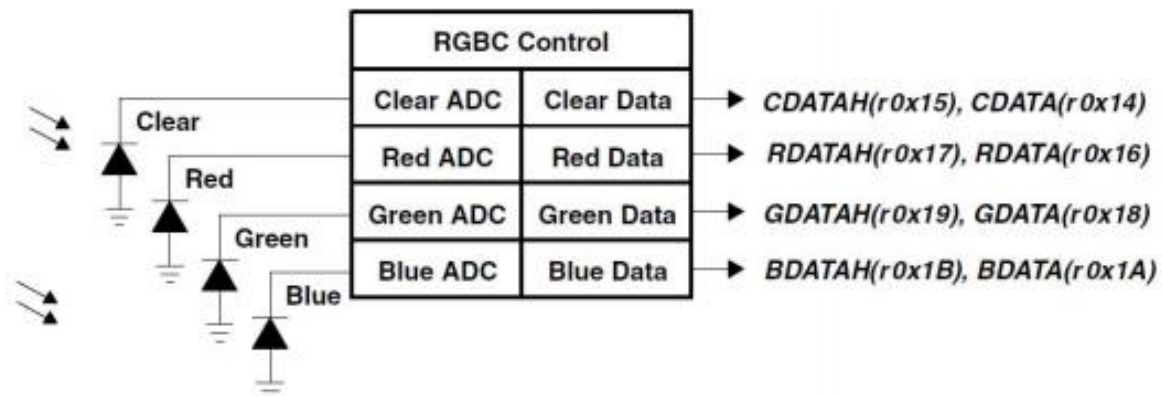
החיישן כולל מערך של 4×3 פוטו-דיודות המשמשים להמרת הצבע לערך אנלוגי. המערך מקבל את האור דרך מסנן אדום, ירוק, כחול וחלקו ללא מסנן.

להלן סכמה של החיישן:



ערכי הצבע שנקראים מרכיבי הפוטו-דיודות נכנסים ל-4 ממירי ADC (ממיר מידע מערך אנלוגי לערך דיגיטלי) בגודל 4 סיביות כל אחד. לאחר המרה מאוחסנים ביחידות זיכרון של הרכיב הנקראים רגיסטרים (אוגרים). ניתן לקרוא את המידע השמור ברגיסטרים ע"י פנייה לכתובות הרגיסטרים באמצעות פרוטוקול I²C.

להלן צילום מסך כתובות הרגיסטרים מתוך דפי נתונים (data sheet) של חיישן הצבע.



12.3 ספריית Adafruit_TCS34725

להפעלת חיישן הצבע נשתמש בספרייה Adafruit_TCS34725. כדי לעבוד עם ספרייה זו יש להצהיר על שימוש בספרייה Adafruit_TCS34725 על ידי שורת הקוד הבאה:

פונקציה tcs.begin

באמצעות פונקציה מבצעים אתחול לחיישן הצבע.

פונקציה tcs.getRawData

קוראים את הערכים מהחיישן ושומרים אותם בכתובות שנשלחו לפונקציה כפרמטרים בהתאמה.

להלן דוגמה לתוכנית שקוראת את הערכים מחיישן הצבע ומציגה אותם על המסך בנוסף בודקת איזה צבע דומיננטי ומציגה את שמו על גבי מסך התקשורת הטורית.

12.4 תכנית דוגמה לזיהוי הצבע

להלן דוגמה לתוכנית שקוראת את הערכים מחיישן הצבע ומציגה אותם על המסך בנוסף בודקת איזה צבע דומיננטי ומציגה את שמו על גבי מסך התקשורת הטורית.

```
#include <Wire.h>
#include "Adafruit_TCS34725.h"
#include <SoftwareSerial.h>
#define LEDsensor 6
byte flag = 0;
Adafruit_TCS34725 tcs =
Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_4X);

void setup() {
  pinMode(LEDsensor, OUTPUT);
  digitalWrite(LEDsensor, LOW);
  Serial.begin(9600);
  Serial.println("Color View Test!");

  if (tcs.begin()) {
    Serial.println("Found sensor");
```

```

    }
    else {
        Serial.println("No TCS34725 found ");
        while (1); // halt!
    }
}

void loop() {
    word clear, red, green, blue;
    unsigned long sum;
    float r, g, b;

    tcs.getRawData(&red, &green, &blue, &clear);
    Serial.print("C:\t"); Serial.print(clear);
    Serial.print("\tR:\t"); Serial.print(red);
    Serial.print("\tG:\t"); Serial.print(green);
    Serial.print("\tB:\t"); Serial.print(blue);

    sum = clear;
    r = (red / sum) * 100;
    g = (green / sum) * 100;
    b = (blue / sum) * 100;
    Serial.print("\t");
    Serial.print((int)r );    Serial.print(" ");
    Serial.print((int)g );    Serial.print(" ");
    Serial.println((int)b );  Serial.println();

    if ((red > green) && (red > blue) && (flag != 1)) {
        flag = 1;
        Serial.println("red");
    }

    else if ((green > red) && (green > blue) && (flag != 2))
{

```

```
flag = 2;

Serial.println("green");

}

else if ((blue > red) && (blue > green) && (flag != 3))
{
    flag = 3;

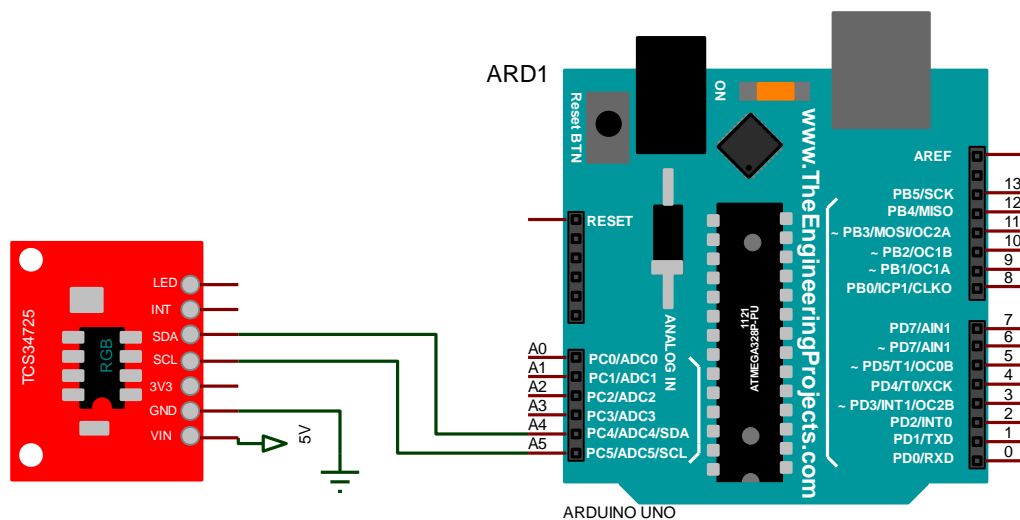
    Serial.println("blue");

}

delay(500);

}
```

להלן שרטוט חשמלי המתאר את חיבור הרכיב אל לוח Arduino:



12.5 כללים לבניית פרויקט

- פתיחה תיקייה לפרויקט
- תכנון המערכת ושרטוט תרשים חשמלי
- הפעלת מרכיבי הפרויקט בנפרד, יש לשמור כל תוכנית בדיקה בנפרד בתיקיית הפרויקט
- אינטגרציה של חלקי הפרויקט יתבצע בשלבים, יש לשלב בהדרגה חלקי תוכניות ולוודא שכל חלק עובד לפני הוספת חלק נוסף.
- כאשר מבצעים אינטגרציה חשוב לשמור תוכניות ראשיות במספור עולה, כאשר תוכנית אינטגרציה עובדת, נשמור אותה לפני שינויים, כך שתמיד נוכל לחזור לשלב הקודם במידה ונתקלנו בבעיה לא פתירה.
- יש לשמור על חיווט מסודר עם צבעים שונים, כך שיקל עלינו לעקוב אחר חיבורים.

12.6 שלבים ל- debug בפרויקט

- הבדיקות מתבצעות מהקל למורכב, אנו ממשיכים לשלב הבא רק אם נפתרה הבעיה.
- יש לעבור על אלגוריתם התוכנה, ולראות האם לה התכוונתם.
- באינטגרציה של תוכניות יש לוודא שהספריות והאובייקטים מוגדרים נכונה (הועברו מתוכנית המקור בצורה מושלמת).
- יש לייצר "דגלים" לאורך התוכנית, במקומות מרכזיים, לבדוק האם הערכים שאתם מקבלים תואמים לציפיות. אפשרות מצוינת להשתמש במסך הסיריאלי.
- יש לוודא שהחומרה מחוברת בצורה תקינה על פי השרטוט החשמלי.
- ניתן להשתמש ברב המודד (ניתן למדוד בו מתח, זרם, התנגדות, רציפות), ובכך להיות בטוחים שההדקים מחוברים ליעדם.
- שימוש במכשירים אלקטרוניים חכמים יותר כמו סקופ למדידת צורת אותות.

13 מקורות

- אתר רשמי של ARDUINO
<https://www.arduino.cc/>
- פיתוח פרויקטים בסביבת Arduino / שי מלול
http://meyda.education.gov.il/files/MadaTech/Megama_Technologit/elecomp/homer-limud/referens-arduino.pdf
- שי מלול , Arduino חלק א' , חלק ב' , הוצאת ידעטק
- sparkfun. פרוטוקולי תקשורת טורית
<https://learn.sparkfun.com/tutorials/i2c/all>
<https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all>
- הסבר מפורט על שני רכיבי הבלוטוס ותכנותם
<http://www.arikporat.com/arduino1/hc-05%20bluetooth.pdf>
- Introduction to MEMS gyroscopes
<https://electroi.com/2010/11/introduction-to-mems-gyroscopes/>
- MEMS Gyroscope Provides Precision Inertial Sensing in Harsh, High Temperature Environments
<https://www.analog.com/en/technical-articles/mems-gyroscope-provides-precision-inertial-sensing.html>
- Servo Motor – Types and Working Principle
<https://www.electronicshub.org/servo-motors/>

14 נספח

14.1 סביבת הפיתוח ב-Arduino

סביבת הפיתוח של Arduino היא סביבה משולבת (IDE – Integrated Development Environment). סביבה זו כוללת עורך קוד (Editor), קומפיילר¹ (Compiler) ובודק שגיאות (Debugger) שמטרתם יצירת סביבה נוחה לפיתוח פרויקטים. תוכנת הצריבה מובנית בסביבת העבודה.

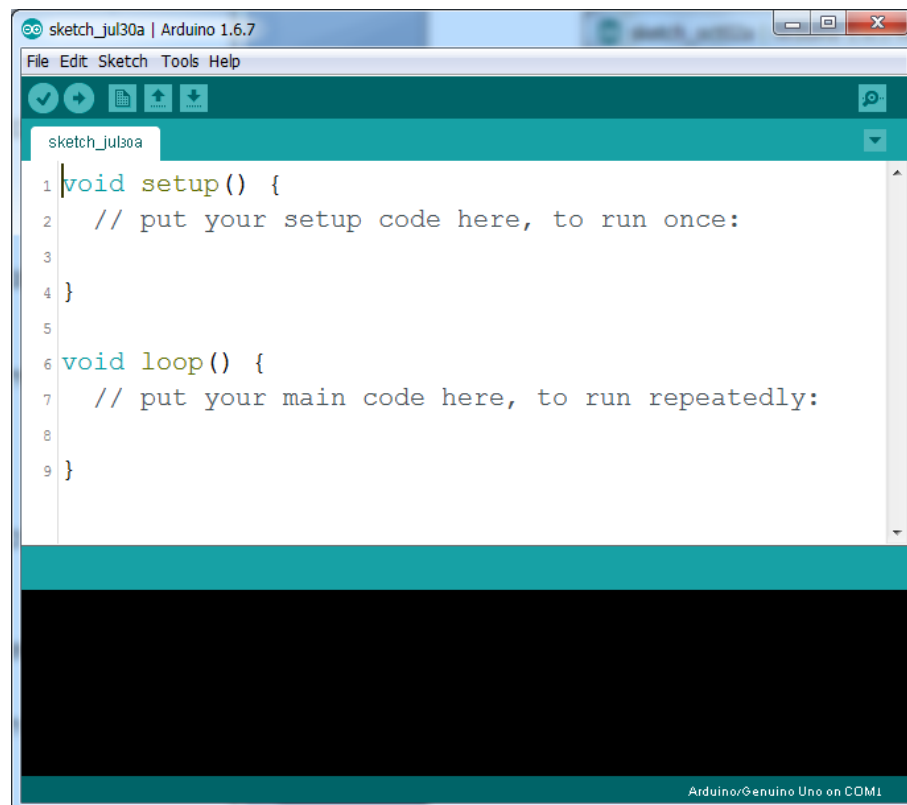
אם אין ברשותכם תוכנת פיתוח של Arduino, הורידו את התוכנה מכאן:

<http://arduino.cc/en/Main/Software>

לאחר הורדת התוכנה והתקנתה ניתן לפתוח אותה ע"י לחיצה על אייקון קיצור הדרך לתוכנה:




כך ייראה החלון שייפתח:





לפניהם האייקונים שמופעים בסרגל הכלים (Toolbar):

¹ מחדר בעברית.



 (Verify) - אייקון המשמש לבדיקת קוד התוכנית. במקרים של טעויות תחביריות בשורות הן יירשמו בחלון התחתון.

 Upload - אייקון המשמש לביצוע קומפילציה מלאה לתוכנית וכן לטעינה/ צריבה של התוכנית למעבד. המעבד כולל תוכנה פנימית המאפשרת קשר עם סביבת הפיתוח במחשב PC.

 New - אייקון המשמש לפתיחת דף חדש ב-Editor (עורך) של התוכנה.

 Open - אייקון המשמש לפתיחת התוכנית.

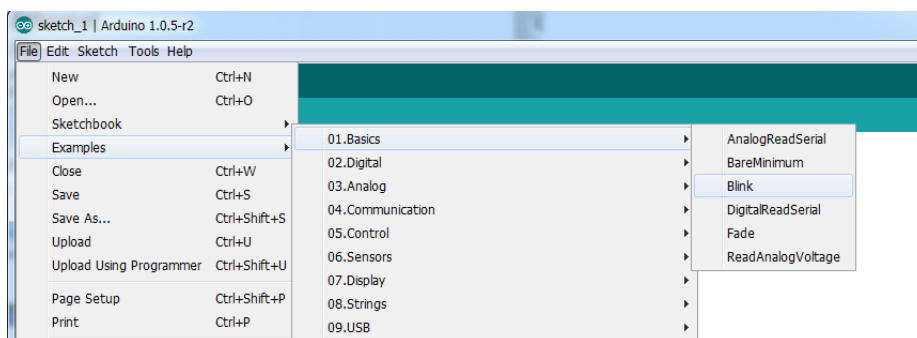
 Save - אייקון המשמש לשמירת התוכנית.

 אייקון המשמש לפתיחת תוכנת Terminal לתקשורת טורית לשימוש כמוניטור.

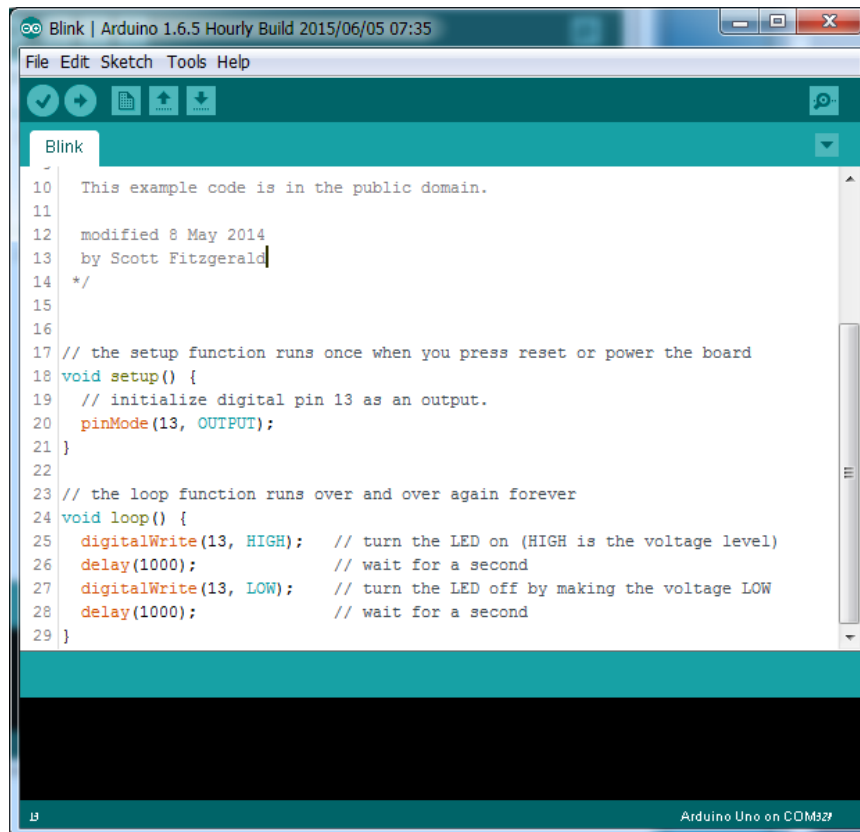
בדיקת הלוח

בתרגיל זה נבצע בדיקה ללוח שברשותכם. לצורך הבדיקה נצרו תוכנית הגורמת לנורת Led L (הממוקמת על הלוח) להבהב כל שנייה. לפתיחת התוכנית נלחץ עם הסמן על לשונית File ונבחר את האפשרות Examples.

בחירה זו תפתח לשונית נוספת שמתוכה יש לבחור באפשרות 01.Basics. מתוך רשימת התוכניות שנפתחה יש לבחור את Blink. להלן צילום מסך העוקב אחר סדר הפעולות שיש לבצע.



זהו החלון שייפתח לאחר בחירה תוכנית Blink:



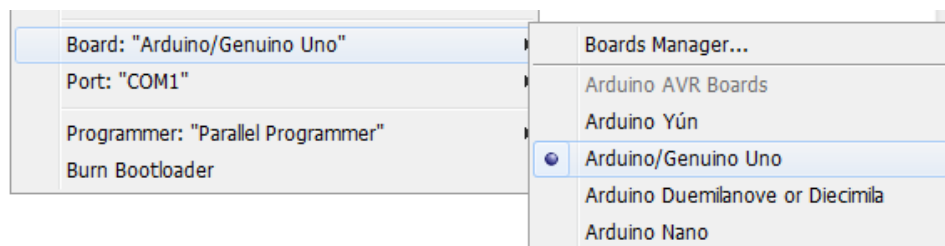
שימו לב: בשלב זה אתם לא אמורים להבין את שורות הקוד המופיעות בתוכנית.

לבדיקת קוד התוכנית לחצו על אייקון  - פעולה זו אמורה להסתיים בהצלחה.

כדי לצרוב את התוכנית למעבד שברשותכם, בצעו את הפעולות האלה:

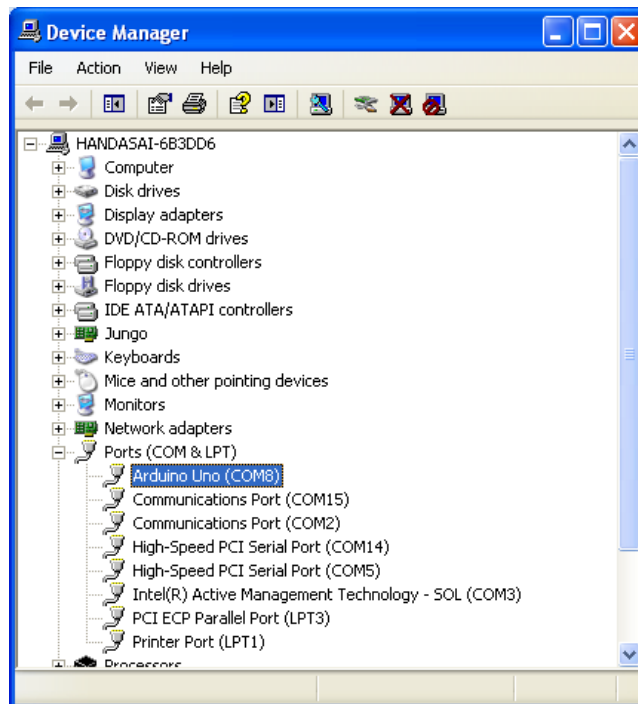
סמנו עם העכבר את לשונית Tools ב-Toolbar, בחרו את אפשרות Board וסמנו מתוך הרשימה שתופיע את האפשרות Arduino Uno.

להלן צילום מסך של הפעולות שעליכם לבצע:



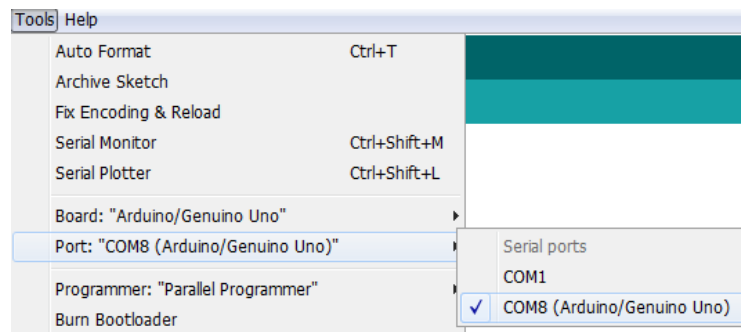
אם הלוח שברשותכם הוא Arduino Nano בחרו באפשרות Arduino Nano/Atmega328. לאחר בחירת הלוח בחרו את יציאת ה-COM שדרכה נצטרך את הלוח. כדי לדעת איזו יציאה מוגדרת ללוח שברשותכם, עליכם לחברו באמצעות כבל USB מתאים אל כניסת ה-USB של המחשב, לפתוח את מנהל ההתקנים ב-Windows ולהתבונן על היציאות.

כך נראה חלון מנהל ההתקנים:



בדוגמה זו ניתן לראות שלוח ה-Arduino במחשב שברשותי מוגדר ביציאה COM8 (ייתכן שבמחשב שלך הלוח יוגדר ביציאה של COM אחר).

סמנו עם העכבר את לשונית Tools, בחרו את האפשרות Serial Ports וסמנו את יציאת ה-COM:



כדי לצרוב את התוכנית לבקר לחצו על האייקון .

זוהי ההודעה שתופיע בחלון שיפתח בסוף תהליך הצריבה:

